Adversary Gain vs. Defender Loss in Quantified Information Flow

Piotr Mardziel,[†] Mário S. Alvim,[‡] and Michael Hicks,[†]
[†]*University of Maryland, College Park*
[‡]*Universidade Federal de Minas Gerais*

***Abstract*—Metrics for quantifying information leakage assume that an adversary's gain is the defender's loss. We demonstrate that this assumption does not always hold via a class of scenarios. We describe how to extend quantification to account for a defender with goals distinct from adversary failure. We implement the extension and experimentally explore the impact on the measured information leakage of the motivating scenario.**

***Keywords*-quantitative information flow, probabilistic models, gain function, vulnerability**

I. Introduction

Quantitative information flow (QIF) is concerned with measuring the amount of secret information that leaks through a system's observable behavior during its execution. The system takes secret (high) input and produces (low) output that can be observed by an adversary. Before the system is run, the adversary is assumed to have some *a priori* information about the secret. As the system executes, the adversary's observations are combined with knowledge about how the system works, resulting in some *a posteriori* information about the secret. A general principle of QIF states that the leakage of information by the execution is defined as the increase in the adversary's information.

Past work has studied how to precisely instantiate this principle, considering various notions of information and how they relate to each other [1], [3]–[5], [8], [9], [16], [23], [27], [30], [32], and increasingly powerful adversaries. For example, active adversaries may be allowed to provide (low) inputs to the system, to manipulate it to leak more data, and adaptive adversaries may choose these inputs based on the observable behavior of the system [14], [17].

Most approaches to QIF consider leakage only from the adversary's point of view, whereas the goals and concerns of the *defender*—i.e., the party interested in protecting information—are overlooked. While in many cases the adversary's gain is directly and inversely related to the defender's loss, this is not always the case. This paper explores this point of view, that is, that the actual leakage of information of an execution is linked to the *defender's loss of secrecy* and not necessarily the *adversary's gain of information*. The following example illustrates this point.

A defender has a stash of gold and a set of 8 possible locations at which to hide it from an adversary, who wants to steal the valuable. Assume that time passes in discrete steps, and that after four time steps the defender can choose to relocate the stash. On the other hand, the adversary can take one of three actions at each time step: (i) choose a location to stake out, incurring in a cost of $\$c$ due to the resources needed for the observation; (ii) choose a location to raid, resulting in a gain of $\$1$ if the gold is found (but no gain otherwise); or (iii) do nothing and wait for another time step, at no cost or gain. The adversary can collect observations or stall for as many time steps as he wants, but once a raid is made the process must stop (perhaps because the defender will realize his gold is at risk and flee the country with it).

We can measure the leakage of the secret in terms of expected value (in $\$$) in knowing the secret. From the adversary's point of view, the leakage relates to his total profit over the process; to maximize leakage, he wants to find the gold with the fewest possible observations. On the other hand, the defender's sole concern is to keep the stash safe, and so leakage for him relates directly to the value ($\$1$) of the stash. Because defender's loss and adversary's gain do not directly oppose each other—in particular, the cost of the adversary's observations is irrelevant to the defender—subtle issues emerge when we try to reason about the leakage of information only from the point of view of the adversary.

As we will show in Section V, if the adversary is rational, his gain can take any value between $\$0$ and $\$1$ depending on observation cost c. However, in the long run there are only two feasible measures of the defender's loss: the gold will be successfully stolen with probability either 1 or 0, and nothing in between is possible. The cutting edge between the two extremes is due to the adversary's interest in maximizing gain: the endeavor will be profitable only if the cost of observing is at most $\$c \leq \$1/7$;[1] otherwise the best strategy is not to observe (or raid) at all. Hence, measuring the threat in terms of adversarial gain could lead one to incorrect conclusions about the security of the defender's secret. This analysis also gives a meaningful insight to the defender's policy making: rather than move the stash more frequently to reduce the risk of theft, he should instead increase the cost of observation to the adversary—anything above $\$1/7$ will make the gold completely safe.

This paper explores a model for QIF that distinguishes the gain of the attacker from the loss to the defender. Our model is a generalization of a model we developed

[1]An analysis that proves this inequality can be found in Section 6.D of our prior paper [18].

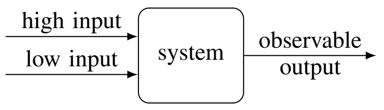


Figure 1. System modeled as an information theoretic channel.

previously that considers *dynamic secrets*, which are secrets the defender may change over time, while interacting with an adaptive (and active) adversary [18]. While our previous model measures leakage only from the adversary's point of view, this paper provides separate metrics for the attacker and defender.

The main contributions of this paper are the following.

- We extend the model of systems with dynamic secrets and adaptive adversaries [18] to distinguish between the defender's loss and adversary's gain. We make explicit the strategy each party adopts and we propose metrics of leakage based on how these two interact.
- We show that both adversary gain and defender loss are necessary to quantify vulnerability of the secret and that if the distinction is ignored, incorrect conclusions can follow.
- We define best- and worst-case measures of loss that describe bounds on loss in the face of the most lucky and unlucky defenders, respectively.
- We also show how sound overestimation of adversary gain can lead to unsound underestimation of defender loss (once again motivating the need to distinguish the two).

The remainder of the paper is organized as follows. Section II reviews the basic information-theoretic approach to quantifying information flow. Section III presents the model we adopt in the paper, which accounts for a defender and an adversary producing inputs to the system according to their own strategies, and Section IV defines metrics of information leakage over this model, and algorithms to compute them. Section V describes a series of experiments involving an implementation of our model that illustrate the above points. Section VI discusses related work, and Section VII concludes.

II. Background

A. The basic information theoretic approach to QIF

The classic model for QIF, pioneered by Denning [10], represents a system as an information-theoretic channel as in Figure 1. The channel is a probabilistic function mapping each high security (i.e., secret) input and low security (i.e., public) input to an observable (i.e., public) output. (High security outputs can also be modeled, but we have no need for them in this work.)

The adversary is assumed to know the probabilistic function describing the channel, and to have some initial *uncertainty* (i.e., a possibly incomplete state of knowledge) about the high input. By providing low input and observing output, the adversary derives some revised uncertainty about the high input. Uncertainty is typically represented with probability distributions [12], and specific metrics of information for QIF map each distribution to a numeric quantity representing the information it contains [6], [8], [9], [25], [30]. Leakage is then defined as the change in the adversary's information.

More formally, let X_{H}, X_{L} and X_{O} be random variables representing the distribution of high inputs, low inputs, and observables, respectively. Given a function $F(X)$ of the information of X, leakage is calculated as follows: [2]

$$F(X_{\mathrm{H}} \mid X_{\mathrm{L}} = \ell, X_{\mathrm{O}}) - F(X_{\mathrm{H}}), \tag{1}$$

where ℓ is the low input chosen by the adversary, $F(X_{\mathrm{H}})$ is the adversary's initial information about the high input, and $F(X_{\mathrm{H}} \mid X_{\mathrm{L}} = \ell, X_{\mathrm{O}})$ is the revised information. As is standard, $F(X_{\mathrm{H}} \mid X_{\mathrm{L}} = \ell, X_{\mathrm{O}})$ is defined to be $\mathbb{E}_{o \leftarrow X_{\mathrm{O}}}[F(X_{\mathrm{H}} \mid X_{\mathrm{O}} = o, X_{\mathrm{L}} = \ell)]$, where $\mathbb{E}_{x \leftarrow X}[f(x)]$ denotes $\sum_x \Pr(X = x) \cdot f(x)$.

Various instantiations of F have been proposed, including Shannon entropy [3], [6], [8], [15], [17], [24], [25], guessing entropy [16], [21], marginal guesswork [27], and (Bayes) vulnerability [5], [30]. In particular, the g-leakage framework [3] uses *gain functions* to model the benefit the adversary obtains by making a particular guess when the secret has a particular value, and define information in terms of adversary gain. Gain functions are expressive enough to model a wide range of scenarios, including when the adversary benefits from guessing part of the secret, guessing a secret approximately, guessing a property of the secret, or guessing the secret within a certain number of tries. As we will see in Section IV, in this paper we will consider notions of information based on a variation of gain functions.

B. System, context, and maximum leakage

In general, the amount of information leaked in an execution varies according to the particular high and low inputs fed to the system. When analyzing the level of security of a system, it is often desirable to obtain robust upper bounds on its leakage that do not depend on any assumption about how exactly inputs will be generated and provided. In order to obtain such bounds, the modeling of the system itself (i.e., the mapping from inputs to outputs) is made independently from the modeling of the *context* in which the system executes, which describes the way inputs are generated and fed to the system.

The *maximum leakage* is an upper bound on the leakage of a system obtained by maximizing the value of leakage

[2]Braun et al. [5] make a distinction between this definition of leakage, called *additive leakage*, and *multiplicative leakage*, where the ratio (rather than the difference) of the a posteriori and a priori information is taken. Divisions by zero avoided, the results of this paper apply to both definitions. For recent results on maximum leakage for both definitions, we refer to Alvim et al. [2].

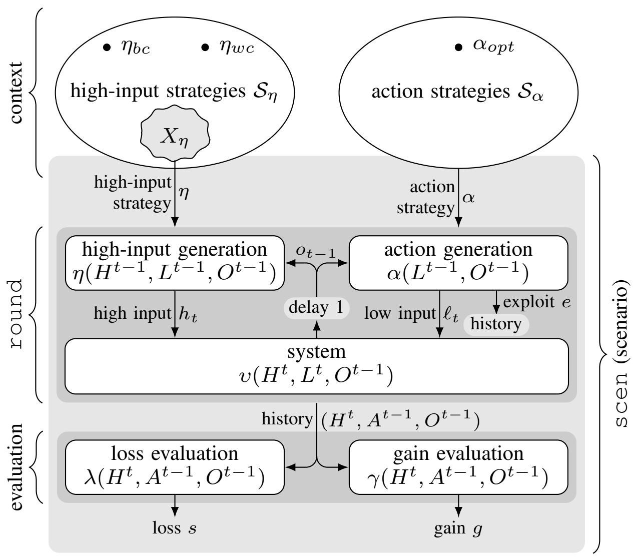

context
η_{bc}
η_{wc}
high-input strategies $\mathcal{S}_\eta$
X_η
α_{opt}
action strategies $\mathcal{S}_\alpha$
high-input strategy η
action strategy α
round
high-input generation $\eta(H^{t-1}, L^{t-1}, O^{t-1})$
o_{t-1}
action generation $\alpha(L^{t-1}, O^{t-1})$
high input h_t
delay 1
low input ℓ_t
exploit e
history
system $\upsilon(H^t, L^t, O^{t-1})$
scen (scenario)
history (H^t, A^{t-1}, O^{t-1})
evaluation
loss evaluation $\lambda(H^t, A^{t-1}, O^{t-1})$
gain evaluation $\gamma(H^t, A^{t-1}, O^{t-1})$
loss s
gain g

Figure 2. Model.

Notation	Explanation
$\mathcal{H}$	Finite set of high inputs
$\mathcal{L}$	Finite set of low inputs
$\mathcal{E}$	Finite set of exploits
$\mathcal{A}$	The set of low inputs and exploits, $\mathcal{L} \cup \mathcal{E}$ (also called actions)
$\mathcal{O}$	Finite set of observables
υ	The system function, of type $\mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{O})$
η	A high-input strategy, of type $\mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{H})$
α	An adversary's action strategy, of type $\mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{A})$
$\mathcal{S}_\eta$	The set of possible high-input strategies
$\Pr(X_\eta)$	The distribution of possible high-input strategies for a given scenario
$\mathcal{S}_\alpha$	The set of possible action strategies
λ, γ	Loss, and gain, functions, respectively, each with type $\mathcal{H}^* \times \mathcal{A}^* \times \mathcal{O}^* \to \mathbb{D}(\mathbb{R})$
$T \in \mathbb{Z}^+$	The latest possible time to attack in a given scenario

Table I
ELEMENTS OF THE MODEL IN FIGURE 2

over all possible configurations of the context of execution. For systems that only accept high inputs, the context can be modeled simply as a distribution on secret inputs, and the maximum leakage is the supremum of the leakage over all possible distributions of secrets. When low inputs are also possible, the description of a context needs to account for both the distribution on secrets and for how adversarial choices of low inputs affect the behavior of the system. In Section IV we will discuss the calculation of maximum leakage for the type of execution we are interested in.

III. Model

This paper extends a model we recently developed to quantify information flow for dynamic secrets (i.e., those that change with time) [18]. The model generalizes the classic model in several ways:

Dynamic secrets: Typical information flow models assume the defender's secret is fixed across adversarial observations. Our model permits the defender to replace the existing secret with a new secret without resetting the measure of leakage; thus from the adversary's point of view, the secret is a *moving target*.

Interactivity: To model a powerful adversary, our model allows *feedback* from outputs to inputs. That is, the adversary can use knowledge gained from one observation to choose a subsequent input to feed into the system.

Input vs. attack: Our model distinguishes inputs that are attacks from those that are not. For example, an adversary might navigate through a website before uploading a maliciously crafted string to launch a SQL injection attack; the navigation inputs themselves are not attacks. This distinction is important when quantifying information flow: the leakage at one point is taken with respect to the expectation of a future attack.

Delayed attack: Just as adversaries are permitted to decide *what* input to provide, they are also permitted to decide *when* is the best time to attack, and this decision process will be considered when quantifying leakage.

This paper extends the model of [18] one step further: when quantifying information leakage from the system, we make the defender's loss distinct from the adversary's gain.

The model is depicted in Figure 2, and consists of roughly three parts: the *context*, at the top; the *round*, in the middle; and the *evaluation*, at the bottom. The latter two phases together are termed the *scenario*. As a whole, the model essentially defines a two-player game with imperfect information and non-zero-sum payoffs. We will limit our analysis to situations where the defender's strategy or the adversary's strategy is fixed. This will let us define metrics by optimizing a party's actions without the need for costly equilibrium analyses. We discuss lifting this limitation in Section VII.

We describe each part of our model in turn, but before we do, we introduce some notation.

A. Notation

- We write $f : \mathcal{A} \to \mathbb{D}(\mathcal{B})$ to designate a *probabilistic* function of type $\mathcal{A} \to \mathcal{B}$, i.e., a function that takes an element of $\mathcal{A}$ as input and probabilistically returns an element of $\mathcal{B}$ (so $\mathbb{D}$ should be read "distribution over").
- Given a random variable X, we will write $x \leftarrow X$ as the process of sampling a value from the distribution $\Pr(X)$ and write $x \in X$ to designate any value x that has non-zero probability according to $\Pr(X)$.
- Strings are given type $\mathcal{S}^* \overset{\text{def}}{=} \{\epsilon\} \cup \mathcal{S} \cup \mathcal{S}^2 \cdots$. We will use capital letters $(L, H, O, \cdots)$ to refer to strings. We will sometimes write S^t to designate an element of $\mathcal{S}^*$ that is t elements long and a_t to refer to the t^{th}

element in some string S. Given $S \in \mathcal{S}^t$ and $a \in \mathcal{S}$ then $S{\cdot}a \in \mathcal{S}^{t+1}$.

The formal terminology that we use is tabulated in Table I.

B. Iterated rounds

The model depicted in Figure 2 characterizes an interaction between two participants, an adversary and a defender. The elements of the classic model from Figure 1 can be seen in the middle part of the figure, the *round*, which is iterated over a series of rounds (hence its name). Here, the defender (the box labeled *high-input generation*) starts by picking an initial secret (high) value out of a set $\mathcal{H}$, and may choose a different secret in subsequent rounds t (the secret value h is indexed by the round t). Likewise, in each round the adversary (the box labeled *action generation*) produces a low input draw from a set $\mathcal{L}$. These inputs are fed into the system υ which produces an observable, visible to both participants, that is drawn from set $\mathcal{O}$. The system may consider past observations as well when producing its result; it has type $\mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{O})$. At some predetermined time T, or before then if it is in his interest, the adversary picks an exploit from a set $\mathcal{E}$, which is then evaluated, along with the history of events so far, to determine the gain of the adversary and the loss of the defender. We discuss the evaluation phase at the end of this section.

The history of an execution can be split into a *secret history* (high values) and a *public history* (low values and observables). The defender's view of an execution consists of both the secret and public history, whereas the adversary's view is restricted to the latter. The choices made in each round by the defender and adversary are specified in terms of *strategy functions*, which depend on their views of the execution. The defender's high-input strategy is a function $\eta : \mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{H})$ that produces the next high value given the history so far. For the adversary, the strategy is a function $\alpha : \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{L} \cup \mathcal{E})$ that produces a low input or an exploit given the public history of the execution. These strategies are determined by the context in a manner described in the next subsection.

Next we characterize the scenario precisely. A single round of interactions is described by the round function (where we write $\mathcal{A}$ to mean the set $\mathcal{L} \cup \mathcal{E}$):

$$\texttt{round}_{\eta,\alpha} : \mathcal{H}^* \times \mathcal{A}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{H}^* \times \mathcal{A}^* \times \mathcal{O}^*)$$

```
1 round$_{\eta,\alpha}$ : $(H, A{\cdot}e, O) \mapsto (H, A{\cdot}e, O)$
2 round$_{\eta,\alpha}$ : $(H, A, O) \mapsto$   // $(A = \epsilon$ or $A \in \mathcal{L}^*)$
3   let $a \leftarrow \alpha(A, O)$ in
4   if $a \in \mathcal{L}$ then
5     let $o \leftarrow \upsilon(H, A{\cdot}a, O)$ in
6     let $h \leftarrow \eta(H, A{\cdot}a, O{\cdot}o)$ in
7       $(H{\cdot}h, A{\cdot}a, O{\cdot}o)$
8   else // $a \in \mathcal{E}$
9     $(H, A{\cdot}a, O)$
```

The subscripts on the function identify the two strategy functions, and the proper arguments characterize the history of high inputs, actions (low inputs or exploits), and observations made so far. Lines 1–8 consider the case that the last action of the adversary was a low input ℓ. In this case, the scenario computes next action of the adversary (line 2) and, if it is a low input (line 3), produces the next observation (line 4) and the subsequent high value using the defender's strategy (line 5). Each of these choices is appended to the existing history (line 6). If the adversary instead produces an exploit (line 7), then just this exploit is added to the history (line 8). An exploit is considered the final event of an interaction, and as such line 9 leaves the history untouched.

C. Context

The context describes the possible strategy functions used in a scenario. It includes a set of high-input strategies $\mathcal{S}_\eta$ and a distribution X_η on these strategies, which encodes an adversary's *prior knowledge* of the possible strategies the defender will use. This distribution helps establish the adversary's initial uncertainty about the defender's secret.

The context also includes a set of adversary strategies $\mathcal{S}_\alpha$ which enumerates the space of behaviors permitted to the adversary. While the defender's strategy can be any one drawn from X_η, our definition of information leakage assumes that adversaries are optimal and pick from the set $\mathcal{S}_\alpha$ only strategies maximizing expected gain. Later, we will define the set $\mathcal{S}_\alpha$ to restrict the adversary in several ways and in so doing show how to define standard QIF metrics.

D. Full scenario and evaluation

The impact of an exploit is evaluated by two functions, as shown at the bottom of the figure. The *gain function* $\gamma : \mathcal{H}^* \times \mathcal{A}^* \times \mathcal{O}^* \to \mathbb{D}(\mathbb{R})$ determines the gain to the adversary; the *loss function* $\lambda : \mathcal{H}^* \times \mathcal{A}^* \times \mathcal{O}^* \to \mathbb{D}(\mathbb{R})$ determines the defender's loss. Notice that gain functions are defined over the *history* of each of the relevant values, rather than, say, the most recent ones. This is because we are quantifying leakage about moving-target secrets, so we must discriminate current and past high values.

To use these functions to compute the information leakage, we can evaluate the scenario up to (at most) some time T and then compute the gain, and loss, of the final outcome. This is done according to the scen function.

$$\texttt{scen}_{\gamma,\lambda} : \mathbb{Z} \times \mathcal{S}_\eta \times \mathcal{S}_\alpha \to \mathbb{D}(\mathbb{R} \times \mathbb{R})$$

```
1 scen$_{\gamma,\lambda}$ : $(T, \eta, \alpha) \mapsto$
2   let $h_0 \leftarrow \eta(\epsilon, \epsilon, \epsilon)$ in
3   let $(H, A, O) \leftarrow$ round$^T_{\eta,\alpha}(h_0, \epsilon, \epsilon)$ in
4   let $s \leftarrow \lambda(H, A, O)$ in
5   let $g \leftarrow \gamma(H, A, O)$ in
6     $(s, g)$
```

Evaluation starts with computing the initial high value h_0 (line 2). It then computes T rounds using round (line

3): on the first iteration it uses arguments $(h_0, \epsilon, \epsilon)$, which produces some output (H^2, A^1, O^1) that is used as input to the next call to `round`, whose output is passed to the next call, and so on; this happens T times with (H, A, O) as the final outcome. The loss s and gain g are computed on lines 4 and 5, respectively.

In the remainder of the paper, when $(s, g) = \mathbb{E}[\texttt{scen}_{\gamma,\lambda}(\cdots)]$ we sometimes write $\texttt{loss}_{\gamma,\lambda}(\cdots)$ and $\texttt{gain}_{\gamma,\lambda}(\cdots)$ to refer to the expected loss s and the expected gain g, respectively.[3]

IV. Metrics

The evaluation component of the model from the previous section produces expected values of loss and gain for given defender and adversary strategies. In this section we use these values to define metrics of the information flow of system executions, and we show how to compute these metrics. Our definitions generalizes several popular metrics from the literature, and Appendix A shows how these metrics can be recovered by simplified variants of our model.

A. Defining the metrics

The amount of information leaked by a system depends on actions performed by both adversary and defender. Our metrics are defined assuming that each party has independent interests, and that each has some knowledge about the other's actions.

In particular, we assume that the adversary knows that the defender draws a high-input strategy according to X_η. Hence an optimal adversary picks an action strategy $\alpha_{opt} \in \mathcal{S}_\alpha$ that would maximize their expected gain. The adversary's *expected gain* $\mathbb{G}$ is defined, then, as the expected gain when using an optimal strategy α_{opt}.

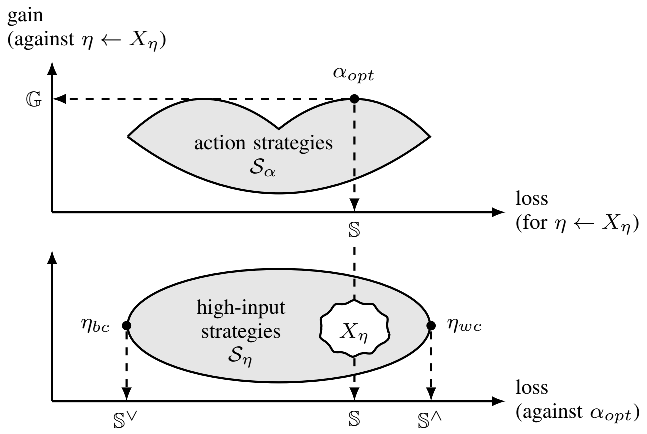
Given that the defender draws a high-input strategy from X_η, the *expected loss* $\mathbb{S}$ is defined as the expectation of the loss evaluation function, assuming the adversary's strategy is their optimal one, α_{opt}.

Figure 3 depicts an example of spaces of strategies, together with the gain and loss they induce. In both graphs, the x-axis represents defender's loss and the y-axis represents adversary gain.

The top graph depicts the set of possible adversary strategies. The maximum expected gain, labeled $\mathbb{G}$, is achieved by an optimal strategy α_{opt}, and this strategy induces an expected loss, labeled $\mathbb{S}$, for the defender. This loss, however, is not necessarily the maximum (or minimum) possible.

Worst- and best-case defender loss are depicted in the bottom graph. They are defined, respectively, as the maximum and minimum expected loss over the set $\mathcal{S}_\eta$ of high strategies when the adversary follows the optimal strategy α_{opt}.

[3]Multivariate expectation $\mathbb{E}[(x, y)]$ is defined as $(\mathbb{E}[x], \mathbb{E}[y])$.



gain (against $\eta \leftarrow X_\eta$)
α_{opt}
$\mathbb{G}$
action strategies $\mathcal{S}_\alpha$
loss (for $\eta \leftarrow X_\eta$)
$\mathbb{S}$
high-input strategies $\mathcal{S}_\eta$
X_η
η_{bc}
η_{wc}
loss (against α_{opt})
$\mathbb{S}^\vee$
$\mathbb{S}$
$\mathbb{S}^\wedge$

Figure 3. Metrics: optimal expected gain (above), and (worst case, best-case, expected) loss (below).

Whereas $\mathbb{S}$ represents the expected loss over all defenders in $\mathcal{S}_\eta$, worst-case loss $\mathbb{S}^\wedge$ and best-case loss $\mathbb{S}^\vee$ are absolute bounds that hold for all defenders.

Before formally introducing metrics to capture the interplay between defender's loss and adversary's gain, we will introduce the following notation.

$\texttt{gain}_{\gamma,\lambda}(T, \eta{\leftarrow}X_\eta, \alpha) \overset{\text{def}}{=} \mathbb{E}_{\eta\leftarrow X_\eta}\left[\texttt{gain}_{\gamma,\lambda}(T, \eta, \alpha)\right]$, and
$\texttt{loss}_{\gamma,\lambda}(T, \eta{\leftarrow}X_\eta, \alpha) \overset{\text{def}}{=} \mathbb{E}_{\eta\leftarrow X_\eta}\left[\texttt{loss}_{\gamma,\lambda}(T, \eta, \alpha)\right]$,

The expectation is taken over the probability $\Pr(X_\eta)$ and the various probabilities in the other model parameters.

Definition 1 (Metrics of information)**.**

1) An adversary's *(optimal) expected gain* against prior $\Pr(X_\eta)$ is the maximal expected gain over the set $\mathcal{S}_\alpha$ of adversary strategies:
$$\begin{aligned}\mathbb{G}^T_\gamma(X_\eta, \mathcal{S}_\alpha, \upsilon) &\overset{\text{def}}{=} \max_{\alpha\in\mathcal{S}_\alpha} \texttt{gain}_{\gamma,\lambda}(T, \eta \leftarrow X_\eta, \alpha)\\ &= \texttt{gain}_{\gamma,\lambda}(T, \eta \leftarrow X_\eta, \alpha_{opt})\end{aligned}$$

2) The defender's *expected loss* against the optimal adversary α_{opt} is defined as:[4]
$$\mathbb{S}^T_{\gamma,\lambda}(X_\eta, \mathcal{S}_\alpha, \upsilon) \overset{\text{def}}{=} \texttt{loss}_{\gamma,\lambda}(T, \eta \leftarrow X_\eta, \alpha_{opt})$$

3) The defender's *worst case* (*best case*) *loss* is the maximal (minimal) expected loss against the optimal adversary α_{opt}:
$$\begin{aligned}\mathbb{S}^{T\wedge}_{\gamma,\lambda}(X_\eta, \mathcal{S}_\eta, \mathcal{S}_\alpha, \upsilon) &\overset{\text{def}}{=} \max_{\eta\in\mathcal{S}_\eta} \texttt{loss}_{\gamma,\lambda}(T, \eta, \alpha_{opt})\\ &= \texttt{loss}_{\gamma,\lambda}(T, \eta_{wc}, \alpha_{opt})\\ \mathbb{S}^{T\vee}_{\gamma,\lambda}(X_\eta, \mathcal{S}_\eta, \mathcal{S}_\alpha, \upsilon) &\overset{\text{def}}{=} \min_{\eta\in\mathcal{S}_\eta} \texttt{loss}_{\gamma,\lambda}(T, \eta, \alpha_{opt})\end{aligned}$$

[4]We assume that if there are multiple optimal adversary strategies, α_{opt} is one which maximizes defender loss among them (that is, uses loss as a tie-breaker in the optimization).

$$= \texttt{loss}_{\gamma,\lambda}(T, \eta_{bc}, \alpha_{opt})$$

(So, η_{wc} and η_{bc} are the strategies that, respectively, maximize and minimize loss against an optimal adversary.)

B. Computing optimal adversary gain and associated defender loss

If the sets of possible strategies are large, enumerating them in order to calculate the given metrics could be too costly. Here we consider the following alternative approach.

We assume that the set of adversary strategies $\mathcal{S}_\alpha$ has a basis of deterministic strategies, which ensures that the search for optimal action strategies can be simplified. We then define a procedure P that predicts the adversary's expected gain and the associated defender's loss. This procedure builds an action strategy (simplified by our assumption) that always picks an action maximizing gain according to the adversary's current belief about the state of the system. This belief is constructed by the procedure `infer` which describes how the adversary updates their knowledge about the high values in an execution, given their view of the system (the public history of adversary actions and observables). This procedure produces a distribution over high values conditioned on the adversary's view. We describe each of these steps in more detail in the rest of this section.

Structure on $\mathcal{S}_\alpha$: We assume the set $\mathcal{S}_\alpha$ has a *basis* composed of deterministic strategies, meaning that whenever an action can be generated by a probabilistic strategy in $\mathcal{S}_\alpha$, there is also a deterministic strategy in $\mathcal{S}_\alpha$ that can generate the same action. We represent this basis in compact form as a *possibilistic strategy* $\alpha_S : \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{P}(\mathcal{A})$, which returns the subset of $\mathcal{A}$ consisting of exactly all actions allowed by some strategy in $\mathcal{S}_\alpha$. The set of all deterministic strategies $\texttt{gen}(\alpha_S)$ generated by the possibilistic strategy α_S is given by

$$\begin{aligned}\texttt{gen}(\alpha_S) \stackrel{\text{def}}{=} \{\alpha \in \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{A}) :\\ \alpha \text{ is deterministic},\\ \alpha \text{ is consistent with } \alpha_S\},\end{aligned}$$

where we say strategy α is *consistent* with possibilistic strategy α_S iff $\text{support}(\alpha(L,O)) \subseteq \alpha_S(L,O)$ for every L, O. [5]

Assuming that $\texttt{gen}(\alpha_S) \subseteq \mathcal{S}_\alpha$, we can greatly reduce the computational cost of finding an optimal strategy, as the best strategy among all of those in $\mathcal{S}_\alpha$ can be found using the possibilistic strategy as the underlying search space. [6]

[5]The support of a distribution $\mathbb{D}(A)$ is the set of elements of A that each have non-zero probability.

[6] This is because in games in which an adversary is optimized against a fixed defender, as is the case here, there is always an optimal deterministic strategy [18], assuming it can choose from among the options available to the probabilistic strategies. Effectively, in general probabilistic strategies can be ignored as long as the space of deterministic ones is general enough (which is the case, for instance, when deterministic strategies are *corner points* for the space of all strategies).

Adversary inference: By interacting with the system and observing the public history of an execution, the adversary learns about the defender's strategy and the high values. For a given context X_η and system υ, this learning process is encapsulated by the procedure `infer` defined below. Given the public history visible to an adversary, the procedure produces a distribution over the secret history conditioned on adversary's view (i.e., consistent with the adversary's observations). We will use this distribution in maximizing the expectation of gain.

$$\texttt{infer} : \mathcal{A}^* \times \mathcal{O}^* \to \mathbb{D}(\mathcal{S}_\eta \times \mathcal{H}^*)$$

```
1 infer:$(\epsilon, \epsilon) \mapsto$
2   let $\eta \leftarrow X_\eta$ in
3   let $h \leftarrow \eta(\epsilon, \epsilon, \epsilon)$ in
4     $(\eta, h)$
5 infer:$(A{\cdot}a, O{\cdot}o) \mapsto$
6   let $(\eta, H) \leftarrow$ infer$(A, O)$ in
7   condition $\upsilon(H, A{\cdot}a, O) = o$ in
8   let $h \leftarrow \eta(H, A{\cdot}a, O{\cdot}o)$ in
9     $(\eta, H{\cdot}h)$
```

The second case (lines 5–9) determines the adversary's knowledge about $H{\cdot}h$ given a non-empty public history $A{\cdot}a$, $O{\cdot}o$. The adversary first determines an initial knowledge about η and H, which is defined recursively (line 6). This knowledge is then refined (via probabilistic conditioning) by taking into consideration that the system function must have returned o when evaluated on $H, A{\cdot}a, O$ (line 7). [7] Finally, he predicts the next high value h as produced from the high strategy η (line 9).

Gain optimization: We can now define P (for "prediction") as the function that, for a given context X_η and system υ, maps the adversary's view to the pair of their optimal expected gain and associated defender's expected loss.

$$P(A,O) \stackrel{\text{def}}{=} \max_{a \in \alpha_S(A,O)} \begin{cases} \mathbb{E}_{\substack{(\cdot,H) \leftarrow \texttt{infer}(A,O) \\ g \leftarrow \gamma(H, A\cdot a, O) \\ s \leftarrow \lambda(H, A\cdot a, O)}} [(g, s)] & (1) \\ \mathbb{E}_{\substack{(\eta,H) \leftarrow \texttt{infer}(A,O) \\ o \leftarrow \upsilon(H, A\cdot a, O)}} [P(A\cdot a, O\cdot o)] & (2) \end{cases}$$

where (1) requires $a \in \mathcal{E}$ or $t = T$, and (2) requires $a \in \mathcal{L}$ and $t < T$, given that $t \stackrel{\text{def}}{=}$ length of A, O. For purposes of maximization, we order pairs lexicographically, favoring g over s, i.e., $(g_1, s_1) < (g_2, s_2)$ whenever $g_1 < g_2$ or $g_1 = g_2$ and $s_1 < s_2$. This is a technicality required for the definition of defender loss when there are multiple optimal adversaries. This definition lets us construct an optimal "demonic" adversary who, when gain is equal, picks actions that maximize loss.

[7]The **condition** statement is akin to an assertion or an *observe* statement found in some probabilistic languages; it serves to probabilistically condition some input distribution on a predicate over its domain.

The definition of P recursively maximizes the expected gain over all actions available to the adversary, given the public history at each time step. Whenever an exploit is produced, or when the maximum time T is achieved, the adversary expected gain is calculated using the result of the gain function evaluated over the public history and the adversary's belief about what the secret history might be. Alternatively, when $t < T$ and the action is a low input, their expected gain is calculated using the gain expected for the adversary's view at the following time step, which includes a prediction of what the next observable might be.

An optimal strategy α_{opt} is defined by replacing $\max$ with argmax in the definition of P above.

Proposition 1. *The recurrence P accurately describes the optimal expected adversary gain and the associated defender loss. Also, the strategy α_{opt} is a strategy that realizes both.*

$$\mathbb{G}_{\gamma}^{T}\left(X_\eta, \mathcal{S}_\alpha, \upsilon\right) = P_1(\epsilon, \epsilon) = \mathtt{gain}_{\gamma,\lambda}(T, \eta \leftarrow X_\eta, \alpha_{opt})$$
$$\mathbb{S}_{\gamma,\lambda}^{T}\left(X_\eta, \mathcal{S}_\alpha, \upsilon\right) = P_2(\epsilon, \epsilon) = \mathtt{loss}_{\gamma,\lambda}(T, \eta \leftarrow X_\eta, \alpha_{opt})$$

C. *Computing worst- and best-case defender loss*

The worst- and best-case defender loss can be determined by computing $\mathtt{loss}_{\gamma,\lambda}(T, \eta, \alpha_{opt})$ for every $\eta \in \mathcal{S}_\eta$. Once again, enumerating the set $\mathcal{S}_\eta$ of defender strategies may be too costly, so we assume additional structure on that set and describe how to find the defender strategies that realize the worst- and best loss.

More precisely, we consider that the defender's strategy can be split into two components: (i) a *controllable* one, which is deterministic and can be directly regulated by the defender, and (ii) an *imposed* one, which can be probabilistic and is beyond the defender's control. As an example, consider a defender who can decide when to reset a secret key (via a controlled component), but does not have control over the value it is reset to (it is picked by the imposed component). This expressiveness is particularly relevant for calculating our best- and worst-case metrics, since it allows us to rule out a defender who, knowing beforehand that the adversary is rational, would behave so to minimize the efficacy of the adversary's strategy (e.g., by always picking the secret key that would be guessed last by the rational adversary). [8]

Formally, the imposed and controllable components of a defender strategy are separated as follows. Define $\bar{\mathcal{H}}$ to be a set of *high actions* (distinct from secret high values $\mathcal{H}$), modeling inputs that the controllable component can produce and feed to the imposed component. A *controllable* half is a strategy $\bar{\eta} : \mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}\left(\bar{\mathcal{H}}\right)$ that produces a high action to be fed to a *imposed* half, which is a function $\eta_f : \mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \times \bar{\mathcal{H}} \to \mathbb{D}\left(\mathcal{H}\right)$ that takes the high action into account to actually produce a high value. A defender strategy, thus, is a composition $\bar{\eta} \circ \eta_f$:

```
$(\bar{\eta} \circ \eta_f) : (H, L, O) \mapsto$
  let $\bar{h} \leftarrow \bar{\eta}(H, L, O)$ in
    $\eta_f(H, L, O, \bar{h})$
```

Our enumeration of high actions is made akin to that of adversary actions in the optimization for gain. We will similarly define a *possibilistic high action strategy* $\bar{\eta}_{\mathcal{S}} : \mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{P}\left(\bar{\mathcal{H}}\right)$, which provides the set of high actions available at any point in a scenario. The search for the worst- and best-case defender loss can be thus restricted to strategies generated as follows:

$$\begin{aligned}\mathtt{gen}(\bar{\eta}_{\mathcal{S}}, \eta_f) \overset{\text{def}}{=} \{\eta \in \mathcal{H}^* \times \mathcal{L}^* \times \mathcal{O}^* \to \mathbb{D}\left(\mathcal{H}\right) :& \\ \eta = \bar{\eta} \circ \eta_f,& \\ \bar{\eta} \text{ is deterministic},& \\ \bar{\eta} \text{ is consistent with } \bar{\eta}_{\mathcal{S}}\}&\end{aligned}$$

We thus assume that $\mathtt{gen}(\bar{\eta}_{\mathcal{S}}, \eta_f) \subseteq \mathcal{S}_\eta$, and that any other strategy in $\mathcal{S}_\eta$ is a composition of some $\bar{\eta}$ (consistent with $\bar{\eta}_{\mathcal{S}}$) and η_f. From here we proceed similarly as in the optimization of adversary behavior. We define $S^\wedge$ and $S^\vee$ as mappings from the defender's view (the full history of the execution) to their expected worst- and best-case loss, respectively.

$$S^\wedge(H, A, O) \overset{\text{def}}{=}$$
$$\max_{\bar{h} \in \bar{\eta}_{\mathcal{S}}(H, A\cdot a, O)} \begin{cases} \mathbb{E}_{\substack{h \leftarrow \eta_f(H, A\cdot a, O, \bar{h}) \\ s \leftarrow \lambda(H\cdot h, A\cdot a, O)}}\left[s\right] & (1) \\ \mathbb{E}_{\substack{h \leftarrow \eta_f(H, A\cdot a, O, \bar{h}) \\ o \leftarrow \upsilon(H\cdot h, A\cdot a, O)}}\left[S^\wedge(H\cdot h, A\cdot a, O\cdot o)\right] & (2) \end{cases}$$

where (1) requires $a \in \mathcal{E}$ or $t = T$, and (2) requires $a \in \mathcal{L}$ and $t < T$, given that $a \overset{\text{def}}{=} \alpha_{opt}(A, O)$ and $t \overset{\text{def}}{=}$ length of H.

The definition of $S^\vee$ for computing the best-case is the same but replacing $\max$ with $\min$ and $S^\wedge$ with $S^\vee$ in the recursive case. Let η_{wc} and η_{bc} be strategies defined by replacing $\max$ and $\min$ above with argmax and argmin respectively, and composing with the imposed η_f as described above.

Proposition 2. *The recurrences $S^\wedge$ and $S^\vee$ accurately describe the worst- and best-case defender loss against the optimal adversary. Also, the strategies exhibiting the worst- and best-case loss are η_{wc} and η_{bc}, respectively.*

$$\begin{aligned}\mathbb{S}_{\gamma,\lambda}^{T\wedge}\left(X_\eta, \mathcal{S}_\eta, \mathcal{S}_\alpha, \upsilon\right) &= S^\wedge(\epsilon, \epsilon, \epsilon) \\ &= \mathtt{loss}_{\gamma,\lambda}(\eta_{wc}, \alpha_{opt}) \\ \mathbb{S}_{\gamma,\lambda}^{T\vee}\left(X_\eta, \mathcal{S}_\eta, \mathcal{S}_\alpha, \upsilon\right) &= S^\vee(\epsilon, \epsilon, \epsilon) \\ &= \mathtt{loss}_{\gamma,\lambda}(\eta_{bc}, \alpha_{opt})\end{aligned}$$

[8]In this paper we do not need impose similar restrictions to the behavior of the adversary, but there may be interesting examples which might require us to do so.

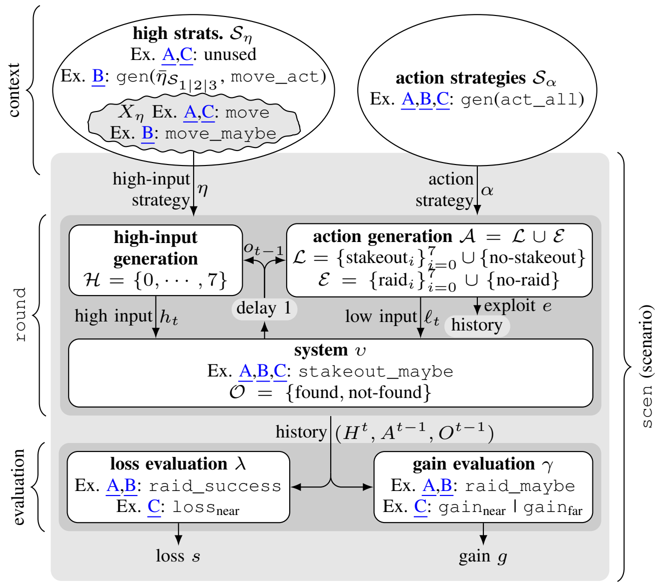

context
high strats. $\mathcal{S}_\eta$
Ex. A,C: unused
Ex. B: gen($\bar{\eta}\mathcal{S}_{1|2|3}$, move_act)
X_η Ex. A,C: move
Ex. B: move_maybe
action strategies $\mathcal{S}_\alpha$
Ex. A,B,C: gen(act_all)
high-input strategy η
action strategy α
round
high-input generation
$\mathcal{H} = \{0, \cdots, 7\}$
o_{t-1}
action generation $\mathcal{A} = \mathcal{L} \cup \mathcal{E}$
$\mathcal{L} = \{\text{stakeout}_i\}_{i=0}^{7} \cup \{\text{no-stakeout}\}$
$\mathcal{E} = \{\text{raid}_i\}_{i=0}^{7} \cup \{\text{no-raid}\}$
high input h_t
delay 1
low input ℓ_t
exploit e
history
system υ
Ex. A,B,C: stakeout_maybe
$\mathcal{O} = \{\text{found, not-found}\}$
scen (scenario)
history (H^t, A^{t-1}, O^{t-1})
evaluation
loss evaluation λ
Ex. A,B: raid_success
Ex. C: loss$_{\text{near}}$
gain evaluation γ
Ex. A,B: raid_maybe
Ex. C: gain$_{\text{near}}$ | gain$_{\text{far}}$
loss s
gain g

Figure 4. Experimental parameters.

V. Experiments

As a proof-of-concept we use a simple implementation to evaluate our model on a few scenarios. We briefly describe the implementation below. In Section V-A we revisit the example from the introduction, analyzing the relation between loss and gain. In the process, we illustrate how to instantiate several parameters of our model that have only been treated abstractly so far. In Section V-B we slightly adapt the example, and analyze worst- and best-case defender loss. Finally in Section V-C we show how sound over-estimation of adversary gain can lead to unsound under-estimation of defender loss. The model parameters we instantiate in our experiments are summarized in Figure 4.

Implementation: We implemented our model using a simple monadic embedding of probabilistic computing [28] in OCaml, as per Kiselyov and Shan [13]. The various probabilistic model parameters are written in monadic style (see [18] for code examples). The recursive procedure P from Section IV-B is computed using these elements to obtain both the optimal adversary gain and associated defender loss, as well as the strategy that achieves these. To compute the worst- and best-case defender loss we similarly implement the recursive procedures $S^\wedge$ and $S^\vee$ of Section IV-C. The implementation (and experiments from the next section) are available online. [9]

A. How does a non-zero-sum utility/gain impact information flow as compared to zero-sum?

Here we analyze our introductory example to show how adversary gain and defender loss are not necessarily aligned and that conclusions from one do not necessarily carry over to the other.

We consider a defender who can choose to hide a stash of gold in one of a set of possible stash locations, modeled as an integer between 0 and 7. At every point in time the adversary may pick one of three actions: (i) stake out, at a cost, a location to learn whether the defender's stash is stored there; (ii) raid a location, obtaining a gain of 1 if the stash is found there, and 0 otherwise; or (iii) stall and not stake out or raid at all (at no gain or cost). Concomitantly, at every 4th time step, the defender can move the stash to a new random location. The defender's concern is only for whether or not the stash stays hidden; the cost incurred by the adversary is of no interest.

This example can be formally captured by our model as follows. (Whenever we use the variable t in the pseudo-code, it will refer to the length of the high portion of the history.)

- Sets of possible high inputs, low inputs, observations, and exploits are encoded as:

$$\mathcal{H} = \{0, \cdots, 7\}$$
$$\mathcal{L} = \{\text{stakeout}_i\}_{i=0}^{7} \cup \{\text{no-stakeout}\}$$
$$\mathcal{O} = \{\text{found, not-found}\}$$
$$\mathcal{E} = \{\text{raid}_i\}_{i=0}^{7} \cup \{\text{no-raid}\}$$

- There is only one possible high-input strategy, move, which at every 4th time step picks a new high value uniformly at random:

```
move : $(H \cdot h, A, O) \mapsto$
  if $t \mod 4 = 0$
    then uniform $\mathcal{H}$
    else $h$
```

  Hence $\mathcal{S}_\eta = \{\texttt{move}\}$. (We are not yet reasoning about worst- and best-case loss, so we do not need to decompose the defender strategy into controllable and imposed components per Section IV-C.)
- The adversary is assumed to know the defender's strategy. The adversary's prior knowledge X_η about possible high-input strategies is encoded as a process taking no arguments and returning high input strategy:

```
$X_\eta$ : $() \mapsto$ move
```

- The possibilistic basis α_S = act_all for the allowable adversary strategies, $\mathcal{S}_\alpha$, allows for any action at any point:

```
act_all : $(A, O) \mapsto \mathcal{L} \cup \mathcal{E}$
```

- The system function, through which the adversary makes observations, is υ = stakeout_maybe, and it determines whether the adversary staked out the correct location or not:

```
stakeout_maybe : $(H \cdot h, A \cdot \text{no-stakeout}, O) \mapsto$
  not-found
stakeout_maybe : $(H \cdot h, A \cdot \text{stakeout}_\ell, O) \mapsto$
  if $h = \ell$ then found else not-found
```

[9] http://ter.ps/fcs14

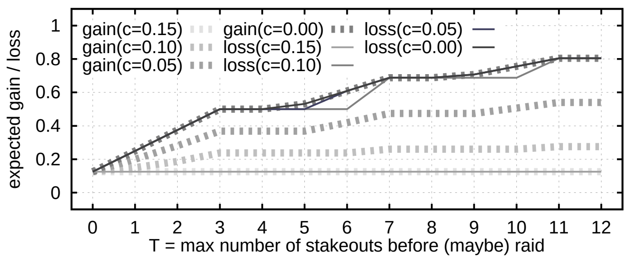


Figure 5. Expected `raid_maybe` gain (dotted lines) and `raid_success` loss (solid lines) with costly stakeouts `stakeout_maybe` and moving stash `move`.

- The adversary gain, γ = `raid_maybe`, computes the cumulative cost incurred for observations made by the adversary. If the adversary decides to raid, the function adds the cost a value of either 1 or 0 depending on whether the raid was, respectively, successful or not successful:

```
raid_maybe:$(H{\cdot}h, A{\cdot}\text{no-raid}, O) \mapsto$
  $-c$ * count_stakeouts($A$)
          // count how many adversary
          // actions were stakeouts
raid_maybe:$(H{\cdot}h, A{\cdot}\text{raid}_\ell, O) \mapsto$
  let raid_gain =
    if $h = \ell$ then 1 else 0 in
  let stakeouts = count_stakeouts($A$) in
    raid_gain - $c$ * stakeouts
```

The process is parameterized by the value c of the cost the adversary incurs for each observation.

- The defender loss, λ = `raid_success`, on the other hand, indicates only the success of the raid if it occurred.

```
raid_success:$(H{\cdot}h, A{\cdot}\text{no-raid}, O) \mapsto 0.0$
raid_success:$(H{\cdot}h, A{\cdot}\text{raid}_\ell, O) \mapsto$
  if $h = \ell$ then 1 else 0
```

Figure 5 demonstrates how adversary gain and defender utility behave in this stakeouts example for varying lengths of time (T). Each line considers a different cost value c: dotted lines show adversary gain, whereas solid ones show defender loss. Lines incurring the same cost have the same color, for easier comparison. Notice that gain very much depends on the cost of observations. With cost 0.00 the gain is highest, and it steadily decreases as cost increases. At 0.15 it is no longer advantageous for the adversary to observe at all, so the gain flattens at $^1/_8$ (their initial gain without any observations). On the other hand, defender loss is not as dependent on the observation cost: it is essentially the same for costs 0.00, 0.05, and 0.10, and it follows in magnitude the gain an adversary would receive were cost equal to 0.00. When the cost is high enough (0.15) the loss drops abruptly to $1/8$, since in this case the adversary would opt to never observe.

From an analytical study of this scenario (see [18]), we know adversary gain and the resulting defender loss can be bounded at 0 as long as the cost for observations is high enough (at least $^1/_7$). For intermediate values of cost, the gain and loss are not identical. There, the gain can be bounded in the limit by $1 - c \cdot {}^1/_7$ but even given this bound, the loss for the defender approaches 1.

B. How can a defender be prevented from a catastrophic worst-case behavior?

This experiment shows how to use worst- and best-case defender loss as a means of quantifying secret vulnerability in the face of, respectively, a catastrophic defender and a particularly lucky one. It shows, on one hand, that a scenario can be sufficiently restricted that even the worst-case defender maintains security for some period of time, and on the other hand, that even the most lucky defender will succumb eventually under those same restrictions.

Best- and worst-case defender loss are extra tools in analyzing the information flow of a system. Though the adversary is expecting to act optimally against a population of defender strategies X_η, here we will look at the range of the resulting loss over three, ever larger, sets of defender strategies $X_\eta \subset \mathcal{S}_{\eta 1} \subset \mathcal{S}_{\eta 2} \subset \mathcal{S}_{\eta 3}$. In effect this provides some guarantees that no matter what strategy is chosen by the defender, their loss will be bounded above and below by the worst case and best case loss. The worst-case is perhaps more relevant as it provides a means of quantifying security even in the face of catastrophic defender behavior.

To illustrate our point, consider a slight modification to the previous example. Assume that now the adversary expects to be attacking a defender who may or may not choose to move the secret stash every 4th time step. To model this variation we decompose the defender strategies into controllable and imposed components, as described in Section IV-C.

Recall that the controllable component produces a high action that the imposed component uses as input when deciding on what high value to generate. Here we define the high actions as follows:

$$\bar{\mathcal{H}} = \{\text{move}, \text{no-move}\} \cup \{\text{move-to}_i\}_{i=0}^{7}$$

In short, the defender can choose to move the stash (but cannot determine where), leave it in place, or move it to a particular location. Assume the adversary thinks the defender's controllable strategy $\bar{\eta}$ considers moving the stash every 4 time steps, but will only do so half the time.

```
decide_move_maybe:$(H, A, O) \mapsto$
  let $c$ ← uniform {true,false} in
    if $t \bmod 4 = 0$ and $c$ then move
    else no-move
```

The imposed component of the defender strategy, η_f, controls the actual movement, and is defined as follows:

```
move_act:$(H, A, O, \text{move}) \mapsto$ uniform $\mathcal{H}$
move_act:$(H{\cdot}h, A, O, \text{move-to}_\ell) \mapsto \ell$
```

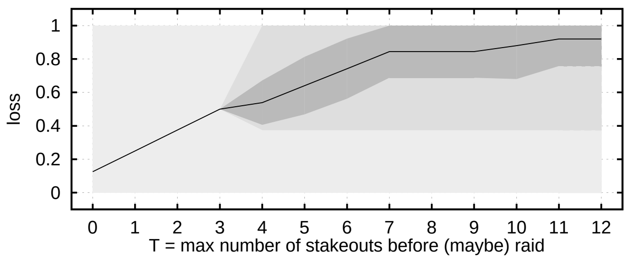


Figure 6. Expected, worst-, and best-case `raid_success` loss against adversary with costly (`cost` = 0.05) stakeouts `stakeout_maybe` and raid `raid_maybe`, with adversary expecting moving stash according to `decide_move_maybe`. Expected loss is solid line whereas the range between worst- and best-case loss are shaded; they are shaded lighter for more permissive defender behaviors.

```
move_act : $(H \cdot h, A, O, \text{no-move}) \mapsto$ h
```

```
move_maybe $\overset{\text{def}}{=}$
  (move_act) ∘ (decide_move_maybe)
$X_\eta : () \mapsto$ move_maybe
```

Though the adversary believes he is interacting with this kind of a defender, we will also analyze the loss with respect to three wider sets of defender strategies, $\mathcal{S}_{\eta 1} \subset \mathcal{S}_{\eta 2} \subset \mathcal{S}_{\eta 3}$. We define the basis functions for these strategies as follows.

For the first, the defender has the control to either change the stash every 4 time steps or not but cannot decide where the stash gets changed to.

```
$\bar{\eta}_{\mathcal{S}_1} : (H, A, O) \mapsto$
  if $t \mod 4 = 0$
    then $\{\text{move}, \text{no-move}\}$ else $\{\text{no-move}\}$
```

The second, wider set of defender strategies, also includes ones in which the defender can move every 4 time steps to anywhere they want but cannot decide where the initial stash is placed:

```
$\bar{\eta}_{\mathcal{S}_2} : (H, A, O) \mapsto$
  if $t = 0$ then $\{\text{move}\}$
  else if $t \mod 4 = 0$
    then $\{\text{move-to}_i\}_{i=0}^{7}$ else $\{\text{no-move}\}$
```

The third, most permissive one, is the set of strategies like the above but also giving the defender control over the initial stash location:

```
$\bar{\eta}_{\mathcal{S}_3} : (H, A, O) \mapsto$
  if $t \mod 4 = 0$
    then $\{\text{move-to}_i\}_{i=0}^{7}$ else $\{\text{no-move}\}$
```

The three sets of strategies are generated from compositions of `move_act` and one of $\bar{\eta}_1$, $\bar{\eta}_2$, $\bar{\eta}_3$, with the extra addition of the sole strategy the adversary believes is employed, `move_maybe`.

$$\mathcal{S}_{\eta 1} \overset{\text{def}}{=} \texttt{gen}(\bar{\eta}_1, \texttt{move\_act}) \cup \{\texttt{move\_maybe}\}$$
$$\mathcal{S}_{\eta 2} \overset{\text{def}}{=} \texttt{gen}(\bar{\eta}_2, \texttt{move\_act}) \cup \{\texttt{move\_maybe}\}$$
$$\mathcal{S}_{\eta 3} \overset{\text{def}}{=} \texttt{gen}(\bar{\eta}_3, \texttt{move\_act}) \cup \{\texttt{move\_maybe}\}$$

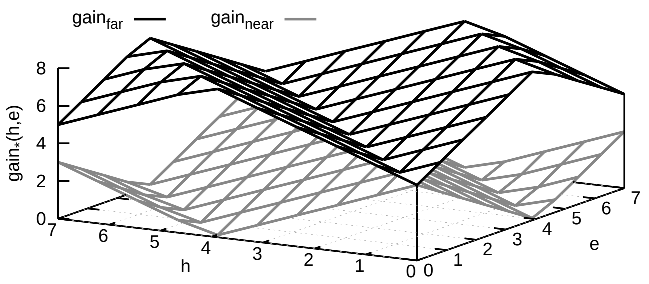


Figure 7. Two gain functions: $\texttt{gain}_{\texttt{near}}$ is inversely proportional to distance between guess and secret, and an over-approximation, $\texttt{gain}_{\texttt{far}}$, is proportional to distance but offset so that $\texttt{gain}_{\texttt{far}} \geq \texttt{gain}_{\texttt{near}}$ on all inputs.

Notice that $X_\eta \subset \mathcal{S}_{\eta 1} \subset \mathcal{S}_{\eta 2} \subset \mathcal{S}_{\eta 3}$. Given the increasing range of defender strategies, the worst- and best-case will likewise increase in the range between them. The results of this experiment can be seen in Figure 6. The line shows the expected loss for the defender drawn from the prior X_η, whereas the shaded regions show the range between worst- and best-case defender loss for varying spaces of defender strategies; lighter regions denote more permissive spaces.

The most permissive set of defender strategies allows the defender to set the initial stash location to the one raided by the adversary at time 0 (before making any observations). This results in the worst-case loss to immediately shoot up to its maximum. On the other hand, the defender can also pick a stash location that the optimal adversary will never check and never raid, keeping the best-case loss at 0.

The less permissive set of strategies prevents the defender from picking the initial stash location. This makes worst- and best-case loss coincide with expected loss until they get a chance to make a best or worst choice at time 4. Then the worst-case shoots up to 1 for the same reason as in the previous case, and the best-case stays near the level it has reached up to that point; the adversary will never get to observe the correct stash location from here on and can only succeed by a lucky raid, as opposed to having some chance to observe the stash while interacting with expected defender (thus the slight drop in defender loss compared to expected loss).

Finally the most restricted set of strategies only allows the defender to choose to move or not move, but not where to move to. In that situation the worst-case is to not ever move, and the best-case is to move every 4 time steps. This results in the worst-case loss increasing faster than the expected and the best-case loss increasing slower than the expected. The expected itself has a 50/50 chance of moving and is approximately between the worst- and best-case loss.

C. How can sound over-estimation adversary gain impact defender loss?

Our final experiment demonstrates the importance of carefully making sure adversary goals are accurately defined if

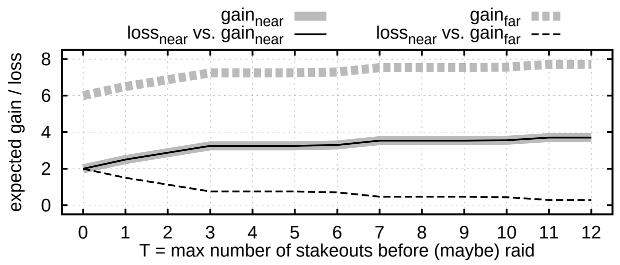


Figure 8. Expected `gain`$_{\texttt{near}}$ gain with resulting (matching) `loss`$_{\texttt{near}}$ loss (solid lines), and expected `gain`$_{\texttt{far}}$ gain with resulting (mismatched) `loss`$_{\texttt{near}}$ loss (dashed lines).

loss is to be correctly quantified. In particular, it is not safe to over-approximate the adversary's gain because doing so may lead to an unsound calculation (i.e., an under-approximation) of loss.

Consider a variation of our introductory example in which the adversary does not need to raid the stash's exact location. Instead of being all-or-nothing, the adversary receives gain proportional to how near the raid is to the stash (assuming stash locations wrap around at 0 and 7 so that distance between 0 and 7 is 1). [10] This is modeled using the following modification to `raid_maybe` function:

```
dist : $(x, y) \mapsto$ min{$|x - y|$, $8 - |x - y|$}

gain$_{near}$ : $(H{\cdot}h, A{\cdot}\text{raid}_e, O) \mapsto$
  let raid_gain = 4 - dist($h, e$) in
  let stakeouts = count_stakeouts($A$) in
    raid_gain - $c$ * stakeouts
```

Thus, the adversary receives gain of 4 when raiding the exact stash location and as little as 0 when raiding a location as far away from the real stash as possible. The defender's loss function `loss`$_{\texttt{near}}$ is identical to `gain`$_{\texttt{near}}$ above.

It may seem at first glance that we can better safeguard the secret by over-estimating the adversary's gain. For example, instead of choosing `gain`$_{\texttt{near}}$, we might assume the adversary receives gain according to a function `gain`$_{\texttt{far}}$ that is greater than `gain`$_{\texttt{near}}$ on all inputs (that is, overestimates adversary gain):

```
gain$_{far}$ : $(H{\cdot}h, A{\cdot}\text{raid}_e, O) \mapsto$
  let raid_gain = 4 + dist($h, e$) in
  let stakeouts = count_stakeouts($A$) in
    raid_gain - $c$ * stakeouts
```

This function returns adversary gain proportional to the distance between the stash and the raid location (instead of inversely). The two gain functions can be seen in Figure 7, for all combinations of stash and raid locations assuming 0 incurred observation costs. The function `gain`$_{\text{far}}$ is plotted in black and is seen above the plot for `gain`$_{\text{near}}$ in gray.

Though the alternate gain function is indeed more conservative from the adversary's point of view, it results in very unsound conclusions about the defender's loss. Figure 8 demonstrates the adversary gain and defender loss under two different scenarios; one in which the gain and loss functions are both `gain`$_{\texttt{near}}$, and one in which the adversary acts to optimize `gain`$_{\texttt{far}}$ instead, whereas the defender still loses according to `loss`$_{\texttt{near}}$. It can be seen there, that indeed the expected gain is higher for `gain`$_{\texttt{far}}$ (thick dashed gray line) than for `gain`$_{\texttt{near}}$ (thick solid gray line) and in that sense `gain`$_{\texttt{far}}$ does result in a more conservative gain calculation. The associated defender loss is not soundly approximated, however. Instead it can be seen that using the `gain`$_{\texttt{far}}$ gain function for the adversary reduces the associated loss (dashed black line) as compared to loss with a `gain`$_{\texttt{near}}$-optimized adversary (solid black line). Essentially, it does not matter that `gain`$_{\texttt{far}}$ is a more conservative gain function than `gain`$_{\texttt{near}}$ in the absolute sense. The relative gains for the adversary encoded in `gain`$_{\texttt{far}}$ make him optimize his actions so that he can raid as far from the stash as possible, the exact opposite of the behavior induced by `gain`$_{\texttt{near}}$.

VI. Related Work

In this paper we adopt the model of Mardziel et al. [18] to model dynamic secrets that evolve over time, and that may vary as the system interacts with its environment. Our model extends that of [18] by distinguishing between adversary's and defender's goals, thus allowing for a more precise quantification of leakage.

Alternative approaches capture interactivity in systems by encoding it as a single "batch job" execution. Desharnais et al. [11], for instance, model the system as a channel matrix of conditional probabilities of whole output traces given whole input traces. Besides creating technical difficulties for computing maximum leakage [1], this approach does not permit low-adaptive or wait-adaptive adversaries, because it lacks the feedback loop present in our model.

O'Neill et al. [26], based on Wittbold and Johnson [31], improve on batch-job models by introducing strategies. The strategy functions of O'Neill et al. are deterministic, whereas ours are probabilistic. And their model does not support wait-adaptive adversaries. So our model of interactivity subsumes theirs.

Clark and Hunt [7], following O'Neill et al., investigate a hierarchy of strategies. *Stream strategies*, at the bottom of the hierarchy, are equivalent to having agents provide all their inputs before system execution as a stream of values. So with stream strategies, adversaries must be non-adaptive. Clark and Hunt show that, for deterministic systems, noninterference against a low-adaptive adversary is the same as noninterference against a non-adaptive adversary. This result does not carry over to quantification of information flow; low-adaptive adversaries derive much more gain than non-adaptive ones, as shown in [18]. At the top of Clark and Hunt's hierarchy, strategies may be nondeterministic, whereas our model's are probabilistic. Probabilistic choice

[10] Say that the pile of gold is so big that it falls into a pyramid-like shape, with a peak in the middle and coins spreading around the area.

refines nondeterministic choice [22], so in that sense our model is a refinement of Clark and Hunt's. But probabilities are essential for information-theoretic quantification of information flow. Clark and Hunt do not address quantification, instead focusing on the more limited problem of noninterference. Nor does their model support wait-adaptive adversaries.

Previous work has considered how the defender's interests can affect the leakage of information in a system. Mardziel et al. [20] consider a defender that, in order to minimize secrecy loss, they can refuse to answer a query involving secret data. They propose a technique to track how accurate the adversary's belief about the secret is, and they implement query analysis and belief tracking via abstract interpretation using probabilistic polyhedra, whose design permits trading off precision with performance while ensuring estimates of a querier's knowledge are sound. In another work, Mardziel et al. [19] use techniques based on belief-tracking to guide multiple defenders in the choice of whether, and to what extend, participate in multiparty computation. Their work considers that each of a set of various defenders holds a secret and must decide whether or not it is safe to cooperate with each other on the computation of a function whose result will be publicly available and may, thus, leak secret information. Rastogi et al. [29] also consider multiparty computations where several agents interact as both defenders and adversaries with respect to each other. They analyze whether automatic optimization of protocols for these computations can produce leakage of information via the state of intermediate variables in protocol execution. None of the works above, however, contemplate scenarios involving both dynamic secrets and active adversaries.

VII. Conclusions and Future Work

In this paper we described how to view quantification of information flow in terms of a game between an adversary and defender with distinct goals. We showed that both adversary gain and defender loss are necessary to quantify vulnerability of the secret and that if the distinction is ignored, incorrect conclusions can follow. We also showed how to use worst- and best-case measures of loss as means of bounding loss in the face of the most catastrophic and most lucky defenders, respectively. We also showed how sound overestimation of adversary gain can lead to unsound underestimation of defender loss.

We have made, however, some strong assumptions as to the knowledge of the adversary and essentially imposed behavior of the defender. It is important in the definitions of gain and loss in our model that only one party optimizes their actions whereas the other's behavior is fixed. When we define gain, we assume the defender's behavior is a strategy sampled from a distribution known to the adversary. Likewise, defender loss uses the adversary strategy that is optimized for this assumption. When we define worst- and best-case defender loss, we instead fix the adversary's strategy and optimize the defender's.

Our simple approach would not work for situations were both parties are attempting to optimize against each other's actions without knowing (or being able to probabilistically predict) them ahead of time. Such situations lead to the notion of *equilibrium*, a pair of defender/adversary strategies η, α such that neither can be changed unilaterally to improve that party's gain (or loss). In some cases equilibria strategies can be found by repeatedly optimizing a strategy against its opponent's until the two converge to a steady state (hence the term "equilibrium"). The examples of this paper, and (we conjecture) essentially any example where the goal of the adversary involves learning the defender's secret, do not exhibit deterministic equilibria; if the defender does not pick their secret randomly, a trivial adversary strategy exists to take advantage of the fact. Our present methods do not let us reason about optimizing non-deterministic strategies.

We are presently working on lifting the deterministic restriction. This would allow us to reason about information flow in terms of equilibria between defender and adversary and free us from the assumption that the adversary knows exactly the distribution over defender strategies.

Acknowledgments: Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] Mário S. Alvim, Miguel E. Andrés, and Catuscia Palamidessi. Quantitative information flow in interactive systems. *Journal of Computer Security*, 20(1):3–50, 2012.

[2] Mário S. Alvim, Konstantinos Chatzikokolakis, Annabelle McIver, Carroll Morgan, Catuscia Palamidessi, and Geoffrey Smith. Additive and multiplicative notions of leakage, and their capacities. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2014. To appear.

[3] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Geoffrey Smith. Measuring information leakage using generalized gain functions. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2012.

[4] Mário S. Alvim, Andre Scedrov, and Fred B. Schneider. When not all bits are equal: Worth-based information flow. In *Proceedings of the Conference on Principles of Security and Trust (POST)*, 2014.

[5] Christelle Braun, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Quantitative notions of leakage for one-try attacks. In *Proceedings of the Conference on Mathematical Foundations of Programming Semantics (MFPS)*, volume 249, pages 75–91, 2009.

[6] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation*, 206(2–4):378–401, 2008.

[7] David Clark and Sebastian Hunt. Non-interference for deterministic interactive programs. In *Proceedings of the Workshop on Formal Aspects in Security and Trust (FAST)*, pages 50–66, 2008.

[8] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative information flow, relations and polymorphic types. *J. of Logic and Computation*, 18(2):181–199, 2005.

[9] Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Quantifying information flow with beliefs. *Journal of Computer Security*, 17(5):655–701, 2009.

[10] Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, Massachusetts, 1982.

[11] Josee Desharnais, Radha Jagadeesan, Vineet Gupta, and Prakash Panangaden. The metric analogue of weak bisimulation for probabilistic processes. In *Proceedings of the Conference on Logic in Computer Science (LICS)*, 2002.

[12] Joseph Y. Halpern. *Reasoning about Uncertainty*. MIT Press, Cambridge, Massachusetts, 2003.

[13] Oleg Kiselyov and Chung-Chieh Shan. Embedded probabilistic programming. In *Proceedings of the Working Conference on Domain Specific Languages (DSL)*, 2009.

[14] Boris Köpf and David Basin. Automatically deriving information-theoretic bounds for adaptive side-channel attacks. *J. Comput. Secur.*, 19(1):1–31, 2011.

[15] Pasquale Malacaria. Assessing security threats of looping constructs. In Martin Hofmann and Matthias Felleisen, editors, *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*, pages 225–235. ACM, 2007.

[16] Pasquale Malacaria. Algebraic foundations for information theoretical, probabilistic and guessability measures of information flow. *Computing Research Repository (CoRR)*, abs/1101.3453, 2011.

[17] Pasquale Malacaria and Han Chen. Lagrange multipliers and maximum information leakage in different observational models. In Úlfar Erlingsson and Marco Pistoia, editor, *Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS)*, pages 135–146, Tucson, AZ, USA, June 2008. ACM.

[18] Piotr Mardziel, Mario Alvim, Michael Hicks, and Michael Clarkson. Quantifying information flow for time-varying data. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2014. To appear.

[19] Piotr Mardziel, Michael Hicks, Jonathan Katz, and Mudhakar Srivatsa. Knowledge-oriented secure multiparty computation. In *Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS)*, 2012.

[20] Piotr Mardziel, Stephen Magill, Michael Hicks, and Mudhakar Srivatsa. Dynamic enforcement of knowledge-based security policies. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2011.

[21] James L. Massey. Guessing and entropy. In *Proceedings of the IEEEInternational Symposium on Information Theory (ISIT)*, 1994.

[22] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Springer, 2005.

[23] Annabelle McIver, Carroll Morgan, Geoffrey Smith, Barbara Espinoza, and Larissa Meinicke. Abstract channels and their robust information-leakage ordering. In *Proceedings of the Conference on Principles of Security and Trust (POST)*, 2014.

[24] Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In *Workshop on Privacy in the Electronic Society 2003*, pages 79–88, 2003.

[25] Ira S. Moskowitz, Richard E. Newman, and Paul F. Syverson. Quasi-anonymous channels. In *Proceedings of the International Conference on Communication, Network, and Information Security (CNIS)*, pages 126–131. IASTED, 2003.

[26] Kevin R. O'Neill, Michael R. Clarkson, and Stephen Chong. Information-flow security for interactive programs. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, pages 190–201, 2006.

[27] John O. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Proceedings of the International Conference in Cryptology in India (INDOCRYPT)*, number 1977, 2000.

[28] Norman Ramsey and Avi Pfeffer. Stochastic lambda calculus and monads of probability distributions. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages (POPL)*, 2002.

[29] Aseem Rastogi, Piotr Mardziel, Matthew Hammer, and Michael Hicks. Knowledge inference for optimizing secure multi-party computation. In *Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security (PLAS)*, 2013.

[30] Geoffrey Smith. On the foundations of quantitative information flow. In *Proceedings of the Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, 2009.

[31] J. Todd Wittbold and Dale M. Johnson. Information flow in nondeterministic systems. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 1990.

[32] Hirotoshi Yasuoka and Tachio Terauchi. Quantitative information flow - verification hardness and possibilities. In *Proceedings of the IEEE Computer Security Foundations Symposium (CSF)*, 2010.

Appendix

A. Expressing existing metrics

Our model here is an extension of the model in [18] which itself extends various models for quantifying information flow. Expressing more limited models is useful to ground the extensions in already familiar frames. We summarize in this section how to express vulnerability, g-vulnerability, and guessing entropy in our model with varying limits on adversary power.

Low-adaptivity: The ability of an adversary to influence the system is modeled using low inputs. This ability can be restricted to model situations without adaptive adversaries by setting $\mathcal{L} = \emptyset$ or alternatively defining adversary strategies such that they ignore observations: $\alpha : (L, O) \mapsto f(L)$. In either way, the adversary lose any ability to adaptively influence the observation process.

Wait-adaptivity: An import element of adversaries introduced in [18] is their ability to adaptively wait for the right time to attack (exploit). This is modeled by an adversary strategy producing an exploit instead of a low input. This power can be restricted by defining adversary strategies so that they only output an exploit at time T and no other time.

Dynamics: Our model allows the secret to change over time based on the history up to that point. We can restrict the model by defining high-input strategies in a manner that lets them pick only the initial high value, but keeps it constant thereafter.

We will use the term *static model* to refer to scenarios in which the adversary and model is restricted as noted above. We then define existing metrics using a static variant of our model.

Vulnerability [30]: The notion of vulnerability corresponds to an equality gain function (i.e., a guess).

```
gain_vul: $(H{\cdot}h, A{\cdot}e, O)$ : $\rightarrow$
  if $h = e$ then 1 else 0
```

The goal of the attacker assumed in vulnerability is evident from `gain_vul`; they are directly guessing the secret, and they only have one chance to do it.

Theorem 2. *In a static model, the vulnerability (written $\mathbb{V}$) of the secret conditioned on the observations is equivalent to dynamic gain using the `gain_vul` gain function.*

$$\mathbb{G}^T_{\mathtt{gain\_vul}}(\cdots) = \mathbb{V}(X_h \mid X_O)$$

In the above we use X_h and X_O to designate the random variables representing the initial/last/only high value and the string of observations, respectively. We also omit the loss function as it does not influence gain.

g-vulnerability [3]: Generalized gain functions can be used to evaluate metrics in a more fine-grained manner, leading to a metric called g-vulnerability. This metric can also be expressed in terms of the static model. Let `gainfunc` be a generalized gain function, taking in the secret and an adversary exploit and returning a real number between 0 and 1, then we have:

```
gain_gen_gain: $(H{\cdot}h, A{\cdot}e, O) \mapsto$
  let $g \leftarrow$ gainfunc $h$ $e$ in $g$
```

The difference between expected gain and g-vulnerability are non-existent in the static model. The gain of a system corresponds exactly to g-vulnerability of `gainfunc`, written $\mathbb{V}_{\mathtt{gainfunc}}(\cdots)$.

Theorem 3. *In a static model the g-vulnerability of `gainfunc` is equivalent to dynamic gain using `gain_gen_gain` gain function.*

$$\mathbb{G}^T_{\mathtt{gain\_gen\_gain}}(\cdots) = \mathbb{V}_{\mathtt{gainfunc}}(X_h \mid X_O)$$

Guessing-entropy [21]: Guessing entropy, characterizing the expected number of guesses an optimal adversary will need in order to guess the secret, can also be expressed in terms of the static model. We let exploits be strings of highs (really permutations of all highs), $\mathcal{E} \overset{\text{def}}{=} \mathcal{H}^{|\mathcal{H}|}$. The exploit permutation corresponds to an order in which the secret is to be guessed. We then define expected gain to be proportional to how early in that string of guesses the secret appears.

```
pos_of: $(h, H) \mapsto$
  // compute the position of h in H //
  // assuming strings are 1-indexed //

gain_guess_ent: $(H{\cdot}h, A{\cdot}e, O) \mapsto$
  let pos $\leftarrow$ pos_of$(h, e)$ in -pos
```

Note that we negate the gain as an adversary would optimize for the minimum number of guesses, not the maximum. Guessing entropy, written $\mathbb{H}_G$, is related to dynamic gain as follows.

Theorem 4. *In a static model, guessing entropy is equivalent to (the negation of) dynamic gain using the `gain_guess_ent` gain function.*

$$-\mathbb{G}^T_{\mathtt{gain\_guess\_ent}}(\cdots) = \mathbb{H}_G(X_h \mid X_O)$$